COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS Versions 5 & 6

September 1994
Vol. 5 No. 9

VERSION 5.0 & 6.x

Your PC really doesn't have a mind of its own

By Van Wolverton

Keeping your system running smoothly and efficiently as it evolves—or sometimes just keeping it running—is both art and science, mixed with a bit of sorcery (and luck). And everyone experiences those times when the only solution is a knowledgeable friend who doesn't mind a midnight call for help.

It doesn't require years of apprenticeship to learn how to configure a PC. We can't cover everything in one article, but we can give you some tips on maintaining your system files as you add hardware and software to your system. We'll also give you some tips on installing new hardware items and their accompanying software. But first, let's look at the differences between Macintosh and DOS platforms.

Contrasting the Macintosh and the PC

Since the Macintosh came along a decade ago, its fans have praised the Mac's usability and mocked PCs in general and the DOS prompt in particular. Those of us with some years of PC experience under our belts occasionally get a bit defensive about these gibes and justifiably point to the continued price-performance advantage of PC-compatible computers and the narrowing usability gap between the two platforms.

But one shortcoming of PCs simply can't be sugar-coated, no matter how badly we'd like to outshine the upstart Mac: Even under the best of circumstances, our systems are a pain to configure. Whenever we change hardware, change software, change our startup procedure, or even just want to make better use of memory, we have to return to the onerous task of making sure DOS knows how to handle the changes we've made.

We have to endure these tedious, time-consuming distractions because the architecture of the PC is an electronic incarnation of anarchy. There are standards, to be sure, such as RS-232 for serial communications and VGA for displays, but nothing in the PC's makeup precludes a manufacturer from offering a better solution. And nothing dictates to us what combinations of hardware and software we can add to our systems.

Contrast this with the Macintosh. If you want a Macintosh computer, you buy it from Apple. Quite a few companies offer hardware accessories and software, but they have to make sure that their products work exactly as the Mac's operating system demands. There aren't any quick-and-dirty ways to get around the operating system and talk directly to the hardware, as you can with a PC. The Macintosh universe is a predictable, orderly one.

So we DOS users reap the benefits of cheaper hardware and wider selection of both hardware and software, but we pay the price of coaxing our systems to keep running when we make changes. Life is full of such trade-offs. However, a little familiarity with CONFIG.SYS and AUTOEXEC.BAT goes a long way toward keeping you in control of your changing system.

Cleaning up CONFIG.SYS

CONFIG.SYS is the file that DOS checks first when you start your system. It contains commands that tell DOS how to manage the hardware that makes up the system. In particular, the configuration commands tell DOS how to manage memory and how to

IN THIS ISSUE

operate devices that are attached to the system, such as CD-ROM drives, scanners, and sound cards. Programs that DOS uses to operate these devices are called device drivers and are identified with the DEVICE and DEVICEHIGH commands.

There is some significance to the order of commands in CONFIG.SYS. In particular, the first three commands should set up the memory management scheme for upper memory (see "Memory Management for Mere Mortals" on page 4). When you install a new item of hardware or software, the installation program may add commands to CONFIG.SYS or modify existing CONFIG.SYS commands.

The installation program may make these changes without warning or let you choose whether to allow it to make the changes (an increasingly common alternative). It's usually best to let the installation program change CONFIG.SYS, but you should periodically check the file yourself with the DOS Editor and do the following:

- If any commands have been added to the beginning of the file, move them past the first three commands that set up memory management.
- If the file contains duplicate commands, delete all but the last one. If the commands are similar but have different values (for example, FILES=65 and FILES=40), delete all but the one that has the value you want.
- Delete any DEVICE command that loads an unnecessary device driver—for example, a piece of hardware no longer attached to your computer or a program that simulates hardware you no longer use. If you aren't sure what driver a DEVICE command refers to, retain the command.

A bit of occasional housekeeping like this keeps you aware of what's in CONFIG.SYS and lets you monitor its contents.

Cleaning up AUTOEXEC.BAT

Installation programs not only tamper with CONFIG.SYS but also add commands to or modify commands in AUTOEXEC.BAT. Here, too, the trend in recent years has been to make installation programs a bit more intelligent and polite. In the first few years after the PC was released, it wasn't unheard of for an installation program to delete your AUTOEXEC.BAT file and replace it with a different one.

There isn't much you can do to protect CONFIG.SYS, other than check it yourself as described earlier, but you can do something to protect AUTOEXEC.BAT from overzealous installation programs: Get rid of it.

Well, don't exactly get rid of it—just change its meaning. Each time you start or restart your system, DOS automatically runs the commands in the file

INSIDE DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices: Domestic: $49/yr ($6.00 each)
Outside US: $69/yr ($8.50 each)

Phone: Toll free: (800) 223-8720
Local: (502) 493-3300
Customer Relations Fax: (502) 491-8050
Editorial Department Fax: (502) 491-3433

Address: You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster: Second class postage is pending in Louisville, KY. Send address changes to
Inside DOS
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in the USA.

named AUTOEXEC.BAT in the root directory of the startup disk. To keep installation programs from altering your startup commands, follow these steps:

1. Change to the root directory and make a copy of AUTOEXEC.BAT named REALAUTO.BAT:

```
cd \
copy autoexec.bat realauto.bat
```

2. Delete all the lines from AUTOEXEC.BAT and replace them with a single command:

```
@realauto
```

Now AUTOEXEC.BAT simply calls the copy of AUTOEXEC.BAT you just created, REALAUTO.BAT. DOS carries out all your startup commands just as before, but AUTOEXEC.BAT itself consists of a single command. After you install a new piece of hardware or software, check what's in AUTOEXEC.BAT. Whatever commands the installation program added will be easy to find. Copy them or make the appropriate changes to the commands in REALAUTO.BAT, keeping the contents of AUTOEXEC.BAT limited to the single command @REALAUTO. (The @ symbol prevents the name of the batch file from displaying during startup.)

Everyone needs an emergency startup diskette

Hard disks are wonderful. It's difficult to appreciate just how wonderful unless you've run a system with nothing but a couple of 360 Kb floppy disk drives. But there's one problem: If something goes wrong, you can't just slip out the hard disk and replace it.

And things do occasionally go wrong. You might even create the problem yourself by changing CONFIG.SYS or AUTOEXEC.BAT so that your system won't start from the hard disk. Or, you might inadvertently delete one of the files that DOS needs to start your system.

What do you do? Start the system from a diskette in drive A, that's what. But in order to do that, you'll

Sometimes computers need a fresh start

By Van Wolverton

Logic tells us that when something goes wrong, we solve the problem by determining the cause and correcting the error. Logic also tells us that a computer is a deterministic system: For each action, there is one and only one response, a response predetermined by the engineers and programmers who designed and built the hardware and software.

Well, yes, that *is* the logical view. Unfortunately, it isn't always accurate, at least not from our vantage point as users of these supposedly deterministic systems. There are times when computers seem not so much creations of Newtonian deterministic physics as members of Einstein's probabilistic universe.

In short, computers don't always seem to follow the rules. Sometimes they fail for no apparent reason; sometimes their behavior seems to violate everything we think we know about cause and effect relationships; and sometimes they seem to be anything but logical machines.

But what's afoot here is the behavior of systems that are more complex than we consumers have ever dealt with before. The hardware may not be particularly complex mechanically—the keyboard has more moving parts than any other part of the system—but the electronic circuitry has the equivalent of millions of transistors. The microprocessor chip alone has more than a million transistors. The software is even more complex—a system of thousands of possible conditions and actions that can interact in virtually unlimited combinations.

Systems can't be fully tested

Occasionally, something goes wrong that may bear no relationship to what you're doing at the moment, something that seems to have no cause whatsoever. There is a cause, but it may be literally impossible to determine. No matter how extensively a company may test its software, there simply is no way to test all possible combinations of actions on all possible combinations of system hardware.

This means that sometimes your computer will seem to exhibit a mind of its own, behaving in a way that seems unrelated to your actions or the commands you give it. Your first reaction may be to panic and start hitting keys in an attempt to undo whatever you did wrong. However, the best way to correct this situation is to shut things down in as orderly a fashion as you can. If possible, save any open files and close any open programs. Then, turn the power off, wait 10 seconds or so, and start the system again. It isn't always sufficient to restart by pressing [Ctrl][Alt][Del], because doing so may alter the contents of memory in a way that can't be corrected merely by restarting DOS.

It's surprising how many times you can fix a problem simply by restarting your system. Yes, this means continuing without understanding exactly what went wrong, but that's one of the prices you pay for using complex systems. All things considered, it's a fairly small price.

have to create an emergency startup diskette by following these steps:

1. Put a diskette in drive A and type the following command to format it as a system disk:

   ```
   format a: /s
   ```

   You can start your system with this diskette in drive A.

2. Type the following commands to change to the DOS directory and copy the necessary command files to the emergency startup diskette:

   ```
   cd \dos
   copy attrib.exe a:
   copy chkdsk.exe a:
   copy debug.exe a:
   copy edit.com a:
   copy fdisk.exe a:
   copy format.com a:
   copy mem.exe a:
   copy qbasic.exe a:
   copy scandisk.exe a:
   copy undelete.exe a:
   copy xcopy.exe a:
   ```

Put this diskette in a safe place. If you're naturally cautious, you might want to make a copy of this diskette with the DISKCOPY command and put each copy in a separate place.

When you install new hardware

Installing a card—for example, a sound card or scanner controller—requires only a Phillips screwdriver and a willingness to remove the case from your computer. Working inside your computer really isn't difficult (it's pretty hard to damage anything inside there), but if the thought of dismembering your system gives you sweaty palms, you should probably have someone else do it.

For a reasonable fee, most retail stores will install hardware devices you purchase from them. Depending on where you live, $25 to $40 should cover almost any straightforward installation, such as a new hard disk or memory upgrade. More complex hardware items, such as a multimedia kit that includes a CD-ROM drive, sound card, and associated software, will probably cost up to $60 or $75.

When you're trying to decide whether it's worth it to have someone else install some hardware for you,

Memory management for mere mortals

By Van Wolverton

When IBM released the PC in 1981, the most common system configuration included 64 Kb of memory. You could buy one with as little as 16 Kb, although you couldn't do much with it. Little wonder, then, that 640 Kb wasn't considered a limitation.

Times quickly changed as application programs and DOS grew, taking up more and more memory. Even though third-party manufacturers quickly brought out add-on cards to bring total memory up to the 640 Kb limit, what once seemed a boundless resource became frustratingly scarce within a few short years.

Other companies came out with programs that let DOS make use of memory above 640 Kb, and DOS gained that capability in Version 5 with the DEVICEHIGH and LOADHIGH commands. Version 6 added MEMMAKER, which tries to automate the process of fussing with CONFIG.SYS and AUTOEXEC.BAT, but memory management still isn't a hands-off job.

A non-mysterious approach

Here's a simple approach that probably won't let you use every possible byte of upper memory but that should significantly increase available conventional memory. (Don't try this unless you have created an emergency startup diskette, as described in the section "Everyone Needs an Emergency Startup Diskette" on page 3.) This approach involves loading device drivers and memory-resident programs into upper memory (sometimes called the *high memory area*).

1. Type the following commands to change to the root directory and make copies of CONFIG.SYS and AUTOEXEC.BAT:

   ```
   cd \
   copy config.sys config.sav
   copy autoexec.bat autoexec.sav
   ```

   You should always make copies of your current system files before you begin to modify them. That way, you have the security of knowing you can return to a working configuration.

2. Issue the MEMORY command by typing *mem*. Note whether the last line of the command output reads *MS-DOS is resident in the high memory area*. Also note the number that follows *Largest executable program size*.

3. Type *edit config.sys* to open CONFIG.SYS in the DOS Editor.

4. If the last line of the MEMORY command output in step 2 was not *MS-DOS is resident in the*

keep in mind that you're saving not only the need to tear into your system, but also the time and aggravation it would take to install the software and make sure everything works OK.

If you do the job yourself, these suggestions can make the job easier:

- Be sure to disconnect all cables from the computer, especially the power cable, before starting any work.
- Give yourself plenty of room. If the computer is on a narrow counter or desk, move it to a larger surface that's at a comfortable working height, such as the kitchen table.
- Get a couple of small bowls or jar lids to hold screws as you remove them.
- Don't rush. Read the instructions and make sure you know what you're going to do before you start removing screws.
- Before putting the case back on, make sure that any cards you added or cables you connected are securely seated in their sockets. Loose connectors are the most common cause of hardware problems.

Hardware usually means software, too

Installing software to support new hardware can range from trivial (copying a file or two to your hard disk) to a real challenge (creating subdirectories, copying files, and modifying CONFIG.SYS or WIN.INI). As when you're installing the hardware, read the instructions carefully and don't enter any commands until you're sure of what you're doing.

After you've installed the software, it's a good idea to check both CONFIG.SYS and AUTOEXEC.BAT to see what changes, if any, the installation program has made. You may find duplicate commands or commands inserted in awkward places (at the very beginning or very end, for example).

If the instructions are unclear (or, worse, incomplete), don't hesitate to call for help. If you purchased the hardware locally, start with the store's customer service or repair department. If they can't—or won't—help you,

high memory area, add or rearrange lines to make sure that the first three lines read as follows:

```
device = c:\dos\himem.sys
device = c:\dos\emm386.exe noems highscan
dos = high,umb
```

(The *noems* switch in the second line prevents access to expanded memory; DOS 6.2x's *highscan* switch tells EMM386 to double-check for available upper memory.)

5. Change all other DEVICE commands in CONFIG.SYS to DEVICEHIGH. Doing so will install these device drivers into upper memory when you start up your computer.
6. Save the changed copy of CONFIG.SYS by choosing Exit from the File menu and pressing [Enter] to save the changes.
7. Type *edit autoexec.bat* to open AUTOEXEC.BAT in the Editor. (If you created REALAUTO.BAT, as we suggested in the lead article on page 1, type *edit realauto.bat* instead.)
8. Add LOADHIGH before each command that runs a memory-resident program, for example:

```
LOADHIGH C:\DOS\SMARTDRV.EXE
LOADHIGH C:\DOS\MSCDEX.EXE /C:MVCD001 /M:10 /V
```

Doing so will load these programs into upper memory when you start your computer.

9. Save the changed copy of AUTOEXEC.BAT by choosing Exit from the File menu and pressing [Enter].
10. Press [Ctrl][Alt][Del] to restart your system. Now type *mem* and note the number that follows *Largest executable program size*. It should be comfortably larger than it was before. You should also see the line that reads *MS-DOS is resident in the high memory area*.

If your system refuses to boot, place your startup diskette in drive A and press [Ctrl][Alt][Del]. Open AUTOEXEC.BAT and CONFIG.SYS in the Editor and correct any errors you find. Or, you can restore the original versions of the files by entering the commands

```
copy config.sav config.sys
copy autoexec.sav autoexec.bat
```

If DOS prompts you for confirmation to overwrite the destination file, respond by pressing Y. Then remove the diskette from drive A and press [Ctrl][Alt][Del] to restart your computer. You're now back where you were before you made the changes, and you still have backup copies of CONFIG.SYS and AUTOEXEC.BAT.

try the manufacturer (and make sure the local store knows that you intend to take your business elsewhere). If the manufacturer is no help, ask a friend with more computer experience than you to give you a hand. As a last resort, take the hardware to the store where you bought it or to a computer repair company (check the phone book; there should be several) and ask them how much they would charge to install it for you. ◆

The fourth edition of *Supercharging MS-DOS*

One of the few prerogatives of a columnist is the occasional plug. Let me, therefore, take this opportunity to tell you that Jeanne and I have embarked on our own publishing venture. Microsoft Press has returned the copyright to *Supercharging MS-DOS* to me. I have revised the book, and Forsyth-Wolf Communications (Jeanne and I) is publishing it.

Don't worry—this isn't some photocopied job sent out in a plain brown wrapper. It's just as handsome as any book you'll buy in a bookstore, and it's full of the same sort of material I cover each month in this column. It costs $25, postpaid (Montana has no sales tax). If you'd like a copy, call (800) 835-2246, extension 68. We accept Visa or MasterCard, or you can send a check for $25 to Forsyth-Wolf Communications, P.O. Box 248, Alberton, Montana, 59820.

—*Van Wolverton*

DOS SHELL TIP

VERSION 5.0 & 6.x

DOSSHELL passwords don't provide much protection

You can use passwords to keep other users from running programs in the DOS Shell's Program List. However, don't be lulled into a false sense of security. Experienced DOS users can circumvent the Shell's passwords with only a few keystrokes.

Even when you assign a DOS Shell password to a program item, such as Editor, QBasic, or Command Prompt, a user can still run the executable file associated with the program item by selecting it in the Shell's File List. For example, if you added a password to the Disk Utilities program group and you use a mouse, a user can still run DEFRAG by clicking the DOS directory in the Directory Tree and then double-clicking the file DEFRAG.EXE in the File List.

If you want to safeguard certain executable files, you can do so by highlighting each filename in the File List and selecting the Change Attributes... option on the File menu. Then select the Hidden option and press [Enter] or click OK. Of course, a knowledgeable user can then reveal all the hidden files by opening the Options menu, selecting the File Display Options..., and then choosing the Display hidden/system files option.

Furthermore, the Shell stores the passwords you assign to program items and groups in uppercase letters in the file DOSSHELL.INI. The Shell doesn't encrypt capital letters in a password at all, and it makes only a minimal attempt to encrypt your password if you use lowercase letters or numbers. Therefore, anyone can highlight the DOSSHELL.INI file in the File List, press [F9], examine DOSSHELL.INI, and decipher all your passwords.

You'll have better luck if you use punctuation marks and other symbols in your passwords because there seems to be no pattern in how the Shell assigns them uppercase letters in the DOSSHELL.INI file. Better yet, mix uppercase and lowercase letters with even a single punctuation mark to keep your potential codebreakers busy for a while. For example, DOSSHELL.INI represents the + sign in a password as the letter *E*. You could change a password such as WEAKARM to W+AKARM, and it would appear in DOSSHELL.INI as WEAKARM. It would take a long time for a user to find which letter—or letters—you replaced with a nonalphabetic symbol.

However, even an indecipherable password doesn't necessarily provide much security: A user can open DOSSHELL.INI in the Editor and simply remove the lines that contain your passwords. If you hide or password-protect the Editor, a user can still exit to DOS and open the Editor by typing *edit*.

Obviously, a truly motivated user can find a way around the Shell's password system. For highly sensitive data, you should use additional data-protection schemes. Also, don't use your passwords from other systems (for example, CompuServe, Prodigy, or MCI Mail) to protect DOS Shell program items, since DOSSHELL stores these passwords in an unprotected file. ◆

BATCH FILE TIP

VERSION 5.0 & 6.x

Speeding up your batch files

If you use lots of remark statements in your batch files, you can make your batch files execute faster by replacing the REM command with a colon or a double colon. In this article, we'll look at how DOS interprets the REM command, the colon, and the double colon in a batch file. Then we'll look at how each of our three comment separators affects a sample batch file's performance.

How DOS interprets REM, :, and :: in a batch file

You use REM in a batch file to document the batch file. For example, you might use the REM command in the first line of this code

```
rem---Add Fred's name at the bottom of each file---
echo Fred J. Smithers > fred.tag
for %%a in (%1) do copy %%a + fred.tag
```

to remind yourself that the commands that follow append Fred Smithers' name to the end of the files you specify when you run the batch file. DOS reads each REM command, so the batch file processes more slowly than it would if the REM lines weren't there.

A colon in a batch file typically identifies a single-word label you can refer to with a GOTO command. For instance, the GOTO command in the first line of this code

```
if "%1"=="\" goto :COMPILE
if not exist %1\nul goto :END
:COMPILE
echo Compiling information for %1 directory
```

sends control to the label :COMPILE in the third line if you enter a backslash as a parameter when you run the batch file. Since the label follows a colon, DOS ignores the label and executes the command on the next line.

Since DOS doesn't execute a label preceded by a colon, you can use the colon to document your batch file. For instance, you could document the batch file in our first example like this

```
:---Add Fred's name at the bottom of each file---
echo Fred J. Smithers > fred.tag
for %%a in (%1) do copy %%a + fred.tag
```

to save DOS the trouble of reading the REM command. As you can see, we didn't include a space after the colon. Omitting the space helps DOS digest the line more quickly, as we'll prove shortly.

Using a colon to introduce a remark statement is a good idea unless the first word of your remark matches a GOTO label. In that case, the batch file might try to execute the commands that follow your remark statement—which could crash your batch file. For this reason, it's probably best to use two colons to introduce a remark in a batch file.

Because the double colon is neither an executable command nor a valid label prefix, DOS can't interpret it and, therefore, ignores it while processing the batch file. Furthermore, when you use a double colon in a batch file, it doesn't matter whether you include a space before your comment. Now let's take a sample batch file out for a spin.

The time trials

We created a batch file that echoes the current time to a text file, runs 500 REM statements, and then calls a second batch file. The second batch file appends the now-current time to the text file and then types that file to the screen.

We ran the batch file five times on a 386, 33 MHz machine. Next, we replaced all the REM commands with colons and ran the batch file again. Then, we ran time trials with a number of variations on the single-colon, double-colon theme. Here are the results:

Form of the remark	*Average speed in seconds*
rem succotash	3.18
: succotash	1.21
:succotash	.99
:: succotash	.99
::succotash	.99

As you can see, replacing the REM command with colons saved more than two seconds on our batch file. Of course, you won't save as much time in a batch file that includes 25 remark statements instead of 500. However, in a complicated batch file, it's important to save every possible second.

For the sake of clarity, we'll continue to use the REM command to introduce remark statements in batch files we run in *Inside DOS*. However, you can use a colon or double colon in your batch files to save processing time. ◆

Misplaced forward slash

In the August 1994 article "Copying Large Directories to More Than One Disk," we inadvertently placed the forward slash (/) for the DIR command's /S switch at the end of the /O-S switch. This error occurs at the top of the second column in Figure A on page 9 and near the top of the second column on page 10. The first segment of the DIR command should read

```
dir %1 /aa /o-s /s /b
```

We're sorry for any inconvenience this caused you!

VERSION 6.2x

Using a COMMAND.COM switch to debug your DOS 6.2x batch files

Do you ever get frustrated when the batch file you just wrote presents you with an error message when you try to run it? Don't you wish DOS would tell you which part of your complicated batch file generates that *Syntax error* or *File not found* message? If this situation sounds familiar and you use DOS 6.2 or higher, you're in luck. In this article, we'll show you a new DOS command switch you can use to help debug your batch files.

COMMAND /Y

DOS 6.2x offers a switch for the COMMAND.COM command that helps you debug your batch files—/Y. You use it in conjunction with the /C or /K switch to step interactively through the batch file. You enter the command like this

```
command /y /c batchfilename
```

or like this

```
command /y /k batchfilename
```

where *batchfilename* is the name of the batch file you want to process one command at a time. You must place the /Y switch before the /C or /K switch.

Running COMMAND.COM in this manner starts a second copy of the command interpreter. The /Y switch directs COMMAND.COM to step through the specified batch file one command at a time. The /C switch tells the command interpreter to perform the specified command—in this case, the batch file—and then exit to the original command interpreter. The /K switch tells the command interpreter to perform the specified command and then display the DOS prompt from inside the second command interpreter.

When you run the command, DOS first asks whether you want to run the batch file, prompting you to answer Y or N. If you press Y, the batch file begins. Then it asks whether you want to run the first command in the batch file, again prompting you to answer Y or N. DOS then presents the next command. The batch file executes each command in its proper logical order—not necessarily in the order the commands appear in the batch file. For example, you might have a batch file that uses a GOTO command to loop back to the top of the batch file. If you press Y when your batch file asks whether you want to execute the GOTO command, the batch file will indeed loop back to the top and ask whether you want to execute the earlier commands again.

An example

To illustrate how this technique works, let's look at what happens when you run the batch file shown in Figure A. This batch file, FUNGO.BAT, contains improper syntax at this point. When you run it, you see this output:

```
Syntax error
You're OUT!
```

The *Syntax error* message tells you something is wrong with the batch file, but it doesn't help pinpoint the problem. Let's use the COMMAND /Y /C command to find where the error occurs and then fix the batch file.

Figure A

```
@echo off
rem FUNGO.BAT illustrates the COMMAND /Y /C command.
for %a in (1 2 3) do echo Strike %a! >> fungo.txt
echo You're OUT! >> fungo.txt
type fungo.txt
del fungo.txt
```

FUNGO.BAT contains improper syntax, as the COMMAND /Y /C command will reveal.

To debug this batch file, you enter the command

```
C:\>command /y /c fungo
```

When you do, DOS presents you with the first prompt:

```
fungo.bat [Y/N]?
```

Answer by pressing Y, and DOS presents the second prompt:

```
echo off [Y/N]?
```

If you answer Y at each prompt, DOS will display, one line at a time, the output shown in Figure B.

Figure B

```
fungo.bat [Y/N]?Y
echo off [Y/N]?Y
rem FUNGO.BAT [Y/N]?Y
for a! >> fungo.txt [Y/N]?Y
Syntax error
echo You're OUT! >> fungo.txt [Y/N]?Y
type fungo.txt [Y/N]?Y
You're OUT!
del fungo.txt [Y/N]?Y
```

You can view the output of the COMMAND /Y /C FUNGO command to find the error in the batch file.

When the batch file types FUNGO.TXT, the text file in the eighth line of the output contains only one line. However, the batch file should have created this text file:

```
Strike 1!
Strike 2!
Strike 3!
You're OUT!
```

As you see, the fifth line of output reads *Syntax error*. This alerts you to look on the fourth line for the problem, and, sure enough, the FOR command doesn't look quite right.

At this point, open FUNGO.BAT in the DOS Editor or another word processor and carefully examine the FOR command. Since you know that this is the faulty command, you can focus all your energies on it. As you look at the command, you notice that you included only a single percent sign (%) before each *a* variable—not two as needed in a FOR command executed from a batch file. Edit the FOR command so that it reads

```
for %%a in (1 2 3) do echo Strike %%a! >> fungo.txt
```

then save the file and exit the Editor.

Now you're ready to debug the batch file again. Enter the command

```
C:\>command /y /c fungo
```

This time if you answer Y at each prompt, DOS will display the output shown in Figure C. As you can see on the 10th through 13th lines in Figure C, the batch file produces the output you want.

Figure C

```
fungo [Y/N]?Y
echo off [Y/N]?Y
rem FUNGO.BAT [Y/N]?Y
for %a in (1 2 3) do echo Strike %a! >> fungo.txt [Y/N]?Y
[Y/N]?Yike 1!
[Y/N]?Yike 2!
[Y/N]?Yike 3!
echo You're OUT! >> fungo.txt [Y/N]?Y
type fungo.txt [Y/N]?Y
Strike 1!
Strike 2!
Strike 3!
You're OUT!
del fungo.txt [Y/N]?Y
```

Corrected syntax makes FUNGO.BAT produce the output you expect.

Notes

We use COMMAND /Y /C to debug a very simple batch file in this example. The command is even more useful for debugging complex batch files. You may want to use the technique we outlined in this article to enhance your understanding of your existing batch files or some of the batch files we present in *Inside DOS*, such as the TODAY.BAT batch file shown on page 11.

Conclusion

If you use DOS 6.2x, you can debug your batch files using a new switch for the COMMAND.COM command. In this article, we showed you how. ◆

A quick way to offer *yes/no* choices in DOS 6.2 batch files

In the accompanying article, we show you how to use the COMMAND /Y /C *batchfilename* command to debug your batch files. When you run this command, DOS presents the batch file one line at a time and asks you to verify whether you want to execute the command on that line. DOS asks you to press Y if you want to execute the command or N if you don't want to execute the command.

You don't have to look too far to see that you can use the same command in a batch file to prompt the user to answer a *yes/no* question. In this case, you use the command in this form

```
command /y /c cmd
```

where *cmd* is the command you want to prompt the user with a *yes/no* choice. For example, you can place in a batch file the command

```
command /y /c copy *.tif b:
```

When you run the batch file and it reaches this command, it will display the prompt

```
copy *.tif b: [Y/N]?
```

and wait for you to press Y or N. If you press Y, the batch file will copy the files to the diskette in the B drive. If you press N, the batch file will execute the command that follows the COPY command.

This technique works with most internal and external DOS commands. However, this technique doesn't work with commands that change environment variables. Also, it doesn't work with the GOTO command, which you use to send control to another section of the batch file. In addition, this technique gives unexpected results with certain commands. For example, if you use the CALL command with the COMMAND command, DOS will prompt you to answer Y or N to each line of the called batch file. Once control returns to the original batch file, you're no longer prompted at each line.

DOSKEY TIP

VERSION 5.0 & 6.x

Using Doskey to stack commands at the DOS prompt

You already know you can use Doskey to return previously entered commands to the command line. In the April 1994 issue, we showed you how to create and save Doskey macros. Well, we have another Doskey tip for you this month: You can use Doskey to enter several commands at once at the DOS prompt. Even if you don't use Doskey for any other purpose, this feature can save you time.

To stack two commands, you simply type the first command, press [Ctrl]T, type the second command, and press [Enter]. DOS displays the character ¶ when you press [Ctrl]T. When you press [Enter], DOS executes the first command and then the second. For example, how many times have you created a directory on a floppy disk and copied files to it from your C drive? To do this, you probably ran two separate commands

```
C:\>md b:\inv_6-93
C:\>copy c:\invoices b:\inv_6-93
```

With Doskey loaded, you can place both commands on a single line

```
C:\>md b:\inv_6-93 ¶ copy c:\invoices b:\inv_6-93
```

Of course, you can use Doskey to stack more than two commands. Simply press [Ctrl]T between the commands and press [Enter] at the end of the line. Your only limitation is the 127-character limit DOS places on any command you type at the prompt.

Let's suppose you want to copy files into several different directories on a floppy disk. To do this from the prompt, you might enter a string of commands like this

```
C:\>copy \invoices b:\inv_6-93
C:\>copy \payroll b:\pay_6-93
C:\>copy \waybill b:\way_6-93
```

You must wait until each set of files copies before you issue the next COPY command. With Doskey loaded, you can enter all the commands on a single line

```
C:\>copy \invoices b:\inv_6-93 ¶ copy \payroll
     b:\pay_6-93 ¶ copy \waybill b:\way_6-93
```

and leave your desk for a few minutes. When you come back, all your files will be copied to the appropriate directory on your backup floppy disk. ◆

LETTERS

Adding the current date and time to the MEMORY.BAT printout

How can I add a date and time to the MEMORY.BAT batch file outlined in the May 1994 article "Tracking the Results of Changes to Your System Configuration"? I'd like to be able to identify each printout chronologically. My experiments using the PROMPT command and the TODAY.BAT batch file from the January 1994 issue of *Inside DOS* have been unsuccessful.

Walter Abbott
La Jolla, California

Great idea! You *can* indeed use TODAY.BAT to add the date and time to the printouts created by MEMORY.BAT. We last visited TODAY.BAT in the January article "Making TODAY.BAT Restore Your Original Directory." The complete batch file is shown in Figure A. (The commands shown in red return you to the directory you were in before you ran the batch file. If you run MEMORY.BAT from the root directory, you don't need these extra commands.)

TODAY.BAT creates memory variables that hold the current date and time. If you want to store TODAY.BAT in a directory other than BATCH, simply change the CD, DIR, and COPY commands to reflect the path to that directory. For example, if you keep your batch files in the root directory, change the CD command in TODAY.BAT to

```
cd \
```

change the COPY command to

```
copy /b c:\today.bat+,, > nul
```

and change the DIR command to

```
dir c:\today.bat | find "TODAY" > tempdate.bat
```

Figure A

```
@echo off
rem TODAY.BAT creates DATE and TIME environment variables.
if not "%4%"=="" goto :SETDATE
echo @prompt cd $p$_$n: > setdirec.bat
command /c setdirec.bat > retdirec.bat
cd \batch
copy /b c:\batch\today.bat+,, > nul
dir c:\batch\today.bat | find "TODAY" > tempdate.bat
tempdate

:SETDATE
set date=%3
set time=%4
del tempdate.bat

call retdirec.bat
del setdirec.bat
del retdirec.bat
```

TODAY.BAT places the current date and time in variables you can use to add a date and time stamp to your printouts.

In order to use TODAY.BAT, you'll need to add to MEMORY.BAT a section that places the contents of the date and time variables on your printout. You can add this section to MEMORY.BAT right after the :REMINDERS section:

```
:ADD_DATE_&_TIME
rem Adds date and time to top of printout.
call c:\batch\today
echo DATE:    %date%        TIME:    %time% > Z.ZZZ
echo. >> Z.ZZZ
```

As before, if TODAY.BAT is in a different directory, add the path to that directory in the CALL command or add that directory to your PATH statement and use the command *CALL TODAY*.

Since you now create the Z.ZZZ text file in the :ADD_DATE_&_TIME section, you need to change the first line of the :COMPILING section from

```
echo CONFIG.SYS file: > Z.ZZZ
```

to

```
echo CONFIG.SYS file: >> Z.ZZZ
```

in order to append this text to the file you already created.

As you can imagine, you can also use TODAY.BAT at the command line to place a date and time stamp at the bottom of any document you print. Once you're ready to print the document, you create the date and time variables by running TODAY.BAT:

```
C:\>today
```

Then, add the date and time stamp to your document by issuing these two commands:

```
C:\>set | find "DATE" >> filename
C:\>set | find "TIME" >> filename
```

where *filename* is the name of the file you're printing. When you print your document, its final two lines will contain the date and time, like this:

```
DATE=09-05-94
TIME=4:07p
```

You must use these commands, rather than the ECHO *%VARIABLE% >> FILENAME* command, to enter the contents of the date and time environment variables into a file from the DOS prompt.

Using Windows Write to change the Editor's default extension from TXT

I frequently use the DOS Editor to create and edit all kinds of text files—batch files, simple databases, letters, etc. Many times, I'll spend half a day at a time in the Editor, modifying or creating one file after another. Very few of my filenames end with the extension TXT, so it's always bothered me that I have to change the file extension in the Open dialog box from TXT to either * or the file extension of the file I want to open. I finally heard about a way to change the default filename extension so that the Editor's Open dialog box will list *.* in the File Name box instead of *.TXT. I thought I'd pass the information along—your readers will appreciate it if they're as frustrated as I was.

The technique uses the Write program that comes with Microsoft Windows to make a small change to the DOS QBASIC.EXE file. First, you make a copy of QBASIC.EXE in case you make a mistake while using Windows Write. Then you open the Windows Write program, which resides in the Windows Accessories group, by double-clicking the icon shown in Figure A.

Figure A

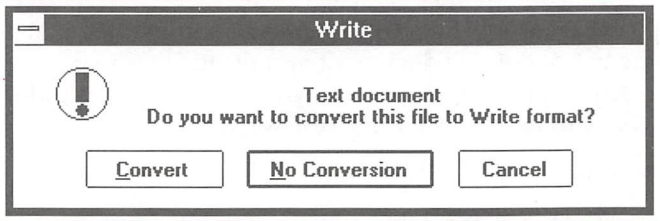
Double-click the Write icon in the Accessories program group to open Windows Write.

Once Write opens, click the File menu and then click the Open... option. In the File Name text box in the Open dialog box, type

```
c:\dos\qbasic.exe
```

and press [Enter]. Next, the dialog box in Figure B will appear—you don't want to convert the file to Write format, so click the No Conversion option.

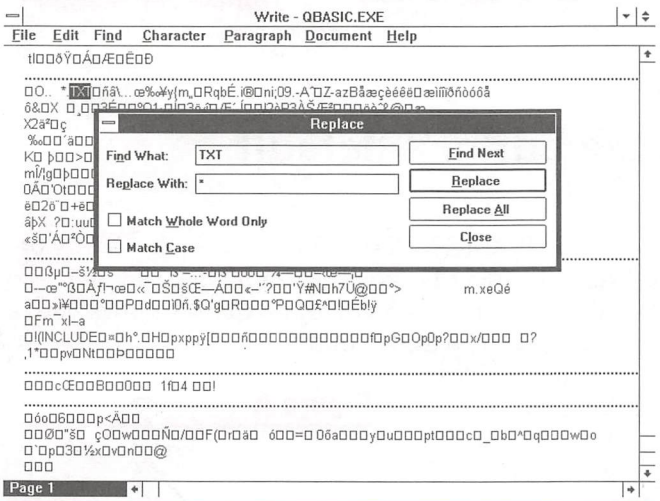
Figure B



You don't want to convert the file's format, so click No Conversion.

The QBASIC.EXE file now opens. At this point, you want to find the text string TXT and replace it with the asterisk followed by two spaces. You *must* include the two spaces or this technique won't work correctly—and you'll risk messing up the entire file. To search for the text string, click the Find menu, click the Replace... option, type *TXT* in the Find What text box, type * and press the spacebar twice in the Replace With text box, and click Replace.

Figure C

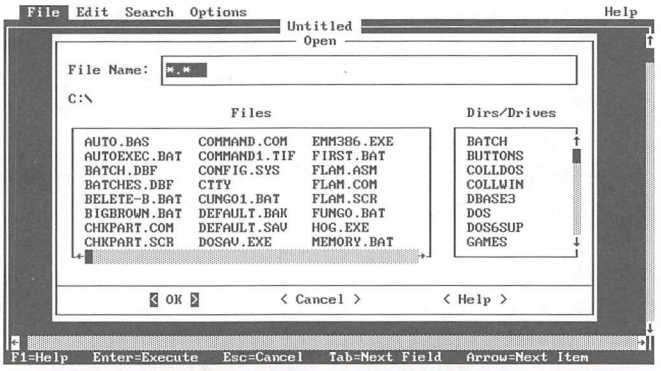


Search for the string TXT *and replace it with an asterisk and two spaces in the Replace dialog box.*

It takes a couple of seconds to find the *.TXT entry in QBASIC.EXE. You can see the Replace dialog box as well as the *.TXT entry in Figure C. When Replace does find the *.TXT entry, click the Replace button again and keep an eye on the highlighted entry in the file. It should display with a couple of trailing spaces. Click OK at the confirmation dialog box, then open the File menu, click Exit, and click Yes when you're asked whether you want to save the changes to QBASIC.EXE.

Now you can exit Windows and make sure that you edited QBASIC.EXE correctly. At the DOS prompt, type *edit* to open the DOS Editor, then click the File menu and click Open.... When the Open dialog box appears, the file specification will appear in the File Name text box as it does in Figure D.

Figure D



*Now the default file extension in the Open dialog box is *, which displays all filenames instead of only the ones ending in* TXT.

Incidentally, if you want to use a different filename extension, such as BAT, you type that extension in the Replace With text box in the Replace dialog box. Be sure to type the new extension in all uppercase letters.

April Campbell
DeKalb, Illinois

This certainly is a useful technique. Thanks very much for passing it along. We can't add anything to improve it—so we won't try. However, we'd like to mention that the editorial staff of *Inside DOS* is always on the lookout for great article ideas. If we use your idea, tip, or technique for an article, we'll give you a byline and a payment ranging from $25 to $100. You can mail your idea, tip, or technique to us at the address listed in the masthead on page 2. Or, if you prefer, you can fax your suggestion to us at (502) 491-3433. ◆